TRAINING REPORT

N.N.2020 - CycleGan - Mario Fiorino 1871233

Premises

Training GAN requires good computational resources and a lot of work to find the "fragile" nash equilibrium to exploit at best the model and produce interesting results.

In my first attempt , in addition to hyperparameter tuning, the principal problem was the value of loss function of discriminator A : it went very close to zero in just few steps (i mean: after 8.000 image processed, loss value d_A < 0.01); this “great accuracy” does not allow to correctly update the generator; the generator may just learn and produce features only to exploit the discriminator!

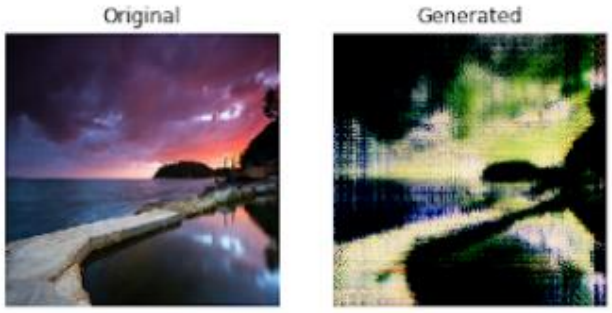
To solve this, i wanted to experiment the idea of putting a random noise on the target values of loss function of discriminator: making it flexible, instead of labeling for every example as 1. Note only on labels of true samples: i.e. the value of target of real samples changes “a little bit” for each epoch:

```
    v_smooth = np.random.uniform(0.9, 1.01)

    pre_smooth_A = (tf.ones_like(decision_r_A)*v_smooth)

    dA_loss_real = tf.losses.mean_squared_error(labels= decision_r_A, predic
tions=pre_smooth_A)
```

This solves the problem of the low values of loss function, but significantly affects the quality of the images, going off target, for examples:



results on 40.000 image processed

So I thought to change strategy, and reflecting on the fact that the dataset A contains only 400 files, i chose to focus on data augmentation techniques. So on input images, first i applied a flipping left/right(with 50% probability to apply or not), then random cropping (50% probability to apply or not), and in the end a Gaussian noise injection (50% probability to apply or not). In the final training version i preferred to remove this last method(noise inject) because it stained the quality of the image generated (than working without) , and since the loss-discriminator problem was solved with just using the flipping and random crooping.
Behind you can see the Tensorboard graphs of the function loss of two discriminator in final and definitive training.

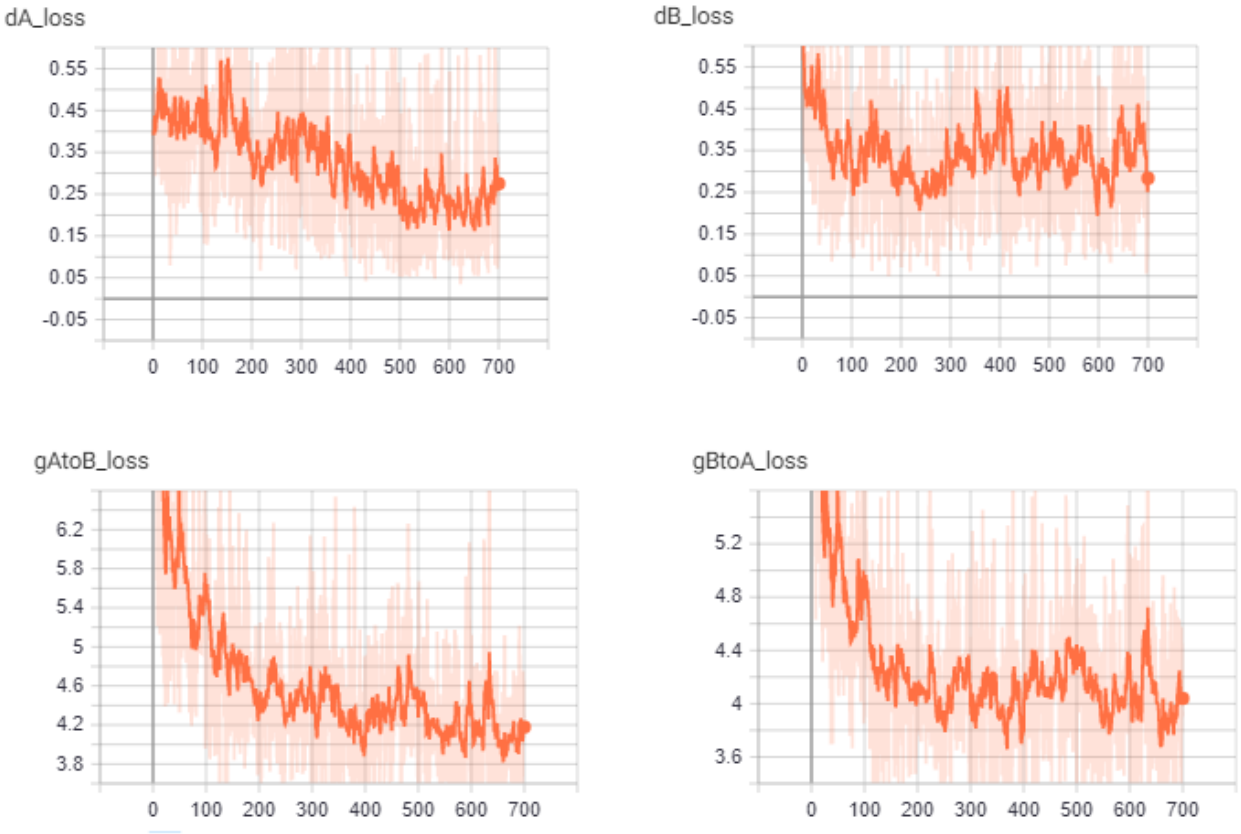
The constrains adversarial and cycle loss are the core of algorithm, but the constrain identity loss don't . So i have experienced to work without. What happens is that the tint (or temperature) of the picture, especially in the phase of reconstruction painting → photo, is not always preserved, often tending on dark blue:

results on 70.000 image processed

I get that this constrain helps (even after applying many filters onto image) to keep the color close to original image. I got the best results to given this loss a multiplication factor of 4.

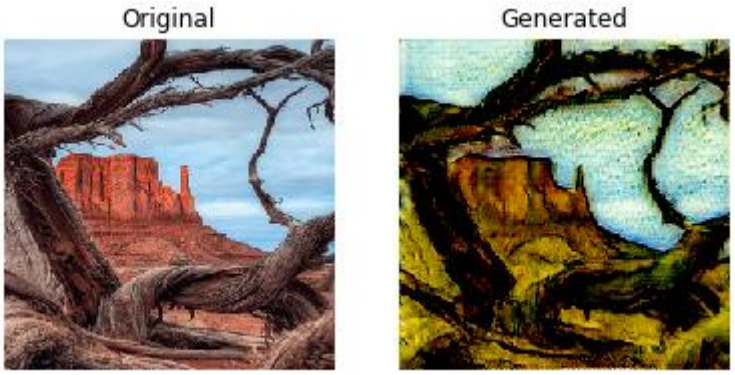
Final Training Results

In each epoch 100 image processed. Total: 70.000 image processed



Neither model has "won". If either the g_loss or the d_loss gets very low it's an indicator that a model is dominating the other. The graphs show that this did not happen, the training was successful. I could have continued the training procedure, but the limited computational resources available to me and the discrete results obtained, made me stop here.

Generated images (input image from dataset test of people.berkeley)



Generated images (input from dataset test personal, note: not only landscape...)